Section 8 Explanations

Contents

1. UNIX, THE OPERATING SYSTEM

1.1. INTRODUCTION

Scope
This chapter describes the basic elements of the UNIX operating system.

Objectives
After studying this chapter, the Philips service engineer has some knowledge of:
how the operating system is organized
the different possibilities in UNIX and is able to:
- manipulate files, directories and devices
- maintain the file system

Prerequisites
In order to understand this chapter the following topics have to be covered:
- Introduction to operating systems
- VAX/VMS or MS-DOS

Multi-users systems

1.2. UNIX OPERATING SYSTEM HISTORY

Designed in 1969, the UNIX system was originally developed for medium-sized minicomputers (DEC PDP series) by Bell Laboratories and later moved to large, powerful mainframe computers as well as microcomputers. Bell Laboratories has consistently introduced new versions of UNIX every few years. There are consequently, many versions of UNIX. Also under the great influence of AT&T, the operating system was further developed to a so called System V.
It is a C-based operating system (Bourne shell).

The university of California at Berkeley introduced a variation of the standard system: the Berkeley Software Distribution (BSD) Version 4.2 using C shell which is a more extended system.
The UNIX version used in SUNOS 4.1 operating system is a heavily enhanced version of the 4.2 and 4.3 BSD UNIX system. It also includes features of AT&T, system V.3 UNIX.
To arrive at a more standardized version of UNIX Sun introduced, in 1993, a new Operating System called "Solaris". Solaris (= SunOS 5.x) is based on the industry standard: UNIX System V Release 4 (SVR4).
Solaris is a total system-software solution, integrating the following:

- OPEN LOOK Window Manager and DeskSet
- Open Windows developer environment

Standardized network protocols.

1.3. UNIX OPERATING SYSTEM FEATURES

The UNIX operating system has the following important features:

Multitasking capability.
Offers the computer the ability to perform several tasks simultaneously. Background tasks are those that can be executed without user's intervention.
Foreground tasks requires user's intervention.

Multi-user capability.
Permits several users to use the same computer simultaneously. More than one user can access the same data at the same time.

Transportability.
Is easier to modify the UNIX system code for installation on a new computer than to rewrite another operating system. More than 90% of the kernel program (explanation follows) is written in C and the rest in machine-specific language. At worse, only 10% of the kernel program must be rewritten to move the kernel to an entirely new machine architecture.

Large selection of powerful UNIX-supplied programs can be divided into two classes:

1. Integral utilities: Parts of the UNIX system that provide such assistance to the operating system that are absolutely necessary for the practical operation of a computer with UNIX. An example is the UNIX system command interpreter, the SHELL program;
2. Tools: Application programs that are not necessary to the computer's basic operation. An example is the word processing "vi".

Communication and electronic mail
Includes the following options:

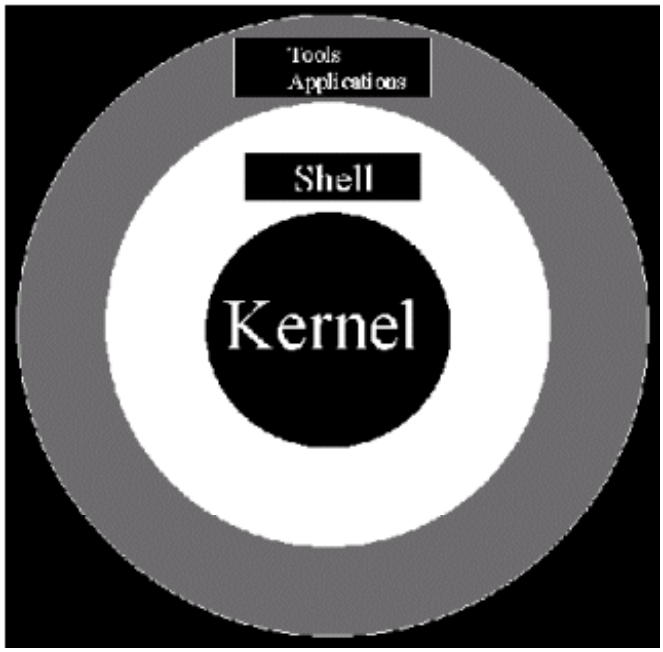
1. Communicating between different terminals hooked into the same computer
2. Communicating between different computers

Library of applications software.
A library of over 500 UNIX applications programs has been developed.

1.4. UNIX OPERATING SYSTEM STRUCTURE

The UNIX system's parts may be functionally categorized into three levels:

- the KERNEL
- the SHELL
- the tools and applications



UNIX operating structure

1.4.1. KERNEL

The kernel is the heart of the operating system, controlling the hardware and actually turning parts of the computer system on and off at program's commands.
The kernel manages all the physical resources. Kernel functions include:

- implements the file system and permits processes to create, read, write, and remove these files
- schedules the tasks, keeps track of all active processes and decides which gets to run next
- manages the device drivers, software routines which control physical devices such as graphics display, mouse, keyboard, disk, tape, serial ports and Ethernet
- supports facilities for creating, examining, and modifying processes
- performs system management functions, such as halting, booting, and error handling
- performs miscellaneous functions which make resources like memory available to processes

The file system, which resides on the disk and provides an organization for all the UNIX system data, is also considered a resource and is managed directly by the kernel. The kernel program resides on the disk in a single file, typically known as vmunix.

1.4.2. SHELL

The shell is a program that connects and interprets the commands typed by a user. It interprets user requests, calls programs from memory, and executes them one at a time or in a series.
The shell program translates the typed commands into commands that the kernel understands.
The C shell (csh), with a C-like syntax, has more useful features for interactive use, such as command aliasing, job control, and a history mechanism.
The Bourne shell (sh), the standard UNIX system command interpreter, while providing fewer interactive features, runs faster and has a simpler syntax for writing shell programs.

1.4.3. TOOLS & APPLICATIONS

The tools and applications level adds special capabilities to the operating system.
Either the CPU is executing the kernel program, a shell program, or another command program. Unix is a time-sharing system.

1.5. THE OPERATING SYSTEM

1.5.1. USER INTERFACE

UNIX uses a three level user interface:
- line orientated
- ASCII full-screen orientated
- X Windows System environment

1.5.2. LOGGING IN & OUT

After pressing RETURN on a terminal which is properly set up, a UNIX system banner line will appear, requesting for a username.
After the UNIX reads your username, it will usually prompt for a password and when both are entered correctly it will display a prompt.

```
Login: username
Password:
EV1#
```

The C shell (standard prompt = # (super user), here represented by EV1#) is ready to accept a command request.
To conclude this first session the command logout or a ^D (Ctrl D) is required. Sometimes the command exit is needed before the logout command.

```
EV1# logout
Login:
```

1.5.3. COMMAND LINE FORMAT

Because in the UNIX system there are several possible options and arguments for most commands, the command line format statement allows us to describe all possible command lines in general terms.

```
EV1# command [option...] [ argument...]
```

The use of brackets ([ ]) around an option or an argument indicates that they are not always required.
The three sequential points (...) indicates that there may be more than one option or argument.
The space character is used as a delimiter.

Examples:

```
EV1# date
EV1# who
```

Remark
The UNIX system distinguishes between uppercase and lowercase letters; an uppercase character is interpreted differently from the lowercase version.

1.5.4. ON-LINE DOCUMENTATION

This UNIX system has the on-line documentation so it is possible to get more information about a particular command. The **man** command diplays information from the reference manuals. It displays complete pages about UNIX subjects.

The following options are supported:

-a Show all manual pages matching name within the MANPATH search path. Manual pages are displayed in the order found.

-d Debug. Displays what a section-specifier evaluates to, method used for searching, and paths searched by man.

-f file… man attempts to locate manual pages related to any of the given files. It strips the leading path name components from each file, and then prints one-line summaries containing the resulting basename or names. This option also uses the windex database.

-F Force man to search all directories specified by MANPATH or the man.cf file, rather than using the windex lookup database. This is useful if the database is not up to date. If the windex data base does not exist, this option is assumed.

-k keyword Print out one-line summaries from the windex database (table of contents) that contain any of the given keywords. The windex database is created using catman(1M).

-l List all manual pages found matching name within the search path.

-M path Specify an alternate search path for manual pages. path is a colon-separated list of directories that contain manual page directory sub-trees. For example, if path is /usr/share/man:/usr/local/man, man searches for name in the standard location, and then /usr/local/man. When used with the - k or - f options, the -M option must appear first. Each directory in the path is assumed to contain subdirectories of the form man*, one for each section. This option overrides the MANPATH environment variable.

-r Reformat the manual page, but do not display it. This replaces the man - -t name combination.

-s section Specify sections of the manual for man to search. The directories searched for name is limited to those specified by section. section can be a digit (perhaps followed by one or more letters), a word (for example: local, new, old, public), or a letter. To specify multiple sections, separate each section with a comma. This option overrides the MANPATH environment variable and the man.cf file. See each Paths below for an explanation of how man conducts its search.

-t man arranges for the specified manual pages to be troffed to a suitable raster output device (see troff(1). If both the - and -t flags are given, man updates the troffed versions of each named name (if necessary), but does not display them.

Some major parts of a man page:

NAME
lists the name or names that are used for the command as well as a brief one-line description of the command's purpose

SYNOPSIS
gives the general command line format for invoking the command

DESCRIPTION
explains in some detail the action of the command

OPTIONS
explains the different options of the command

FILES
if present, lists the file or files that are associated with the command program

BUGS
Basic commands UNIX system documentation

Examples:

This demonstrates this facility, assume information is requested about the command “man”.

```
EV1# man man
Reformatting page.  Wait... done
```

A full example is given for the command “pwd”.

```
EV1# man pwd
Reformatting page.  Wait... done
```

NAME
pwd - return working directory name

SYNOPSIS
/usr/bin/pwd

AVAILABILITY
SUNWcsu

DESCRIPTION
pwd writes an absolute path name of the current working directory to standard output.

Both the Bourne shell, sh(1), and the Korn shell, ksh(1), also have a built-in pwd command.

ENVIRONMENT
See environ(5) for descriptions of the following environment variables that affect the execution of pwd: LC_MESSAGES and NLSPATH.

EXIT STATUS
The following exit values are returned:
0 Successful completion.
>0 An error occurred.

If an error is detected, output will not be written to standard output, a diagnostic message will be written to standard error, and the exit status will not be 0.

SEE ALSO
cd(1), ksh(1), sh(1), shell_builtins(1), environ(5)

DIAGNOSTICS
``Cannot open .." and ``Read error in .." indicate possible file system trouble and should be referred to a UNIX system administrator.

NOTES
If you move the current directory or one above it, pwd may not give the correct response. Use the cd(1) command with a full path name to correct this situation.

1.5.5. OVERVIEW CONTROL KEYS

Key Function:

Ctrl-U erases entire command line
Ctrl-W erases last word on command line
Ctrl-C interrupts many programs and shell scripts
Ctrl-Z suspends many programs and shell scripts
Ctrl-S sops output of running program; prevents output from running off end of screen
Ctrl-Q resumes output from program stopped by Ctrl-S
Ctrl-O throws away output from program without interrupting the program
Ctrl-D End-Of-File character used for logout; also terminates file input

1.5.6. FILE SYSTEM & TYPES OF UNIX FILES

The UNIX file system is a structure for organizing information or data. The data is grouped into named entities called files. The UNIX system recognizes at least three types of files:

1. Ordinary files_This type of file is used to store data. Executable programs (commands) are also stored as ordinary files.
2. Directory files A directory file contains a list of files.
3. Special files_These files are used to reference physical devices, such as terminals, printers, disks, and tape drives. They are read from and written to, just like ordinary files, but such requests cause activation of the associated physical device.

1.5.7. FILE ACCESS PERMISSIONS

There are three classes of file users and three modes of file access offering the UNIX file system a total of nine different kinds of access permission.

The three classes of system users are:

1. **Owner -** The owner is usually the system user who created the file. The super-user can change the individual ownership of a file if necessary. The owner has full control over restricting or permitting access to the file at any time.
2. **Group -** A system user who is not the file owner may access the file if this user belongs to the group of system users who are allowed to access the file. However, this user cannot restrict or permit access to the file; only the owner is allowed to do this.
3. **Other -** This category refers to any other user of the system.

There are three ways of accessing a file. The meaning of these access modes is somewhat different for ordinary files than it is for directories:

1. **Read -** Allows examination of contents Allows listing of files within directory
2. **Write -** Allows changing contents of file Allows creating new files and removing old ones
3. **Execute -** Allows executing file as command Allows searching directory

1.5.8. FILE FIELDS

File field have the following structure:

```
-rwxrwxrwx  1 root  dev    10876  May 16 9:42 part2
```

Where:

"-	The dash in the file type field indicates that the file is an ordinary file.
" Other possibilities:	
D	the entry is a directory;
L	the entry is a symbolic link;
b	the entry is a block special file;
c	the entry is a character special file;
p	the entry is a fifo (or "named pipe") special file;
-	the entry is an ordinary file;
|	the entry is a FIFO.
"rwxrwxrwx"	The permissions are indicated as follows:
r	the file is readable
w	the file is writable
x	the file is executable
-	the indicated permission is not granted

Some examples of a file's permissions are:

```
-rwxr--r--
```

This describes a file that is readable, writable, and executable by the user and readable by the group and others.

```
-rwsr-xr-x
```

This describes a file that is readable, writable, and executable by the user, readable and executable by the group and others, and allows its user-ID to be assumed, during execution, by the user presently executing it.

```
-rw-rwl---
```

This describes a file that is readable and writable only by the user and the group and can be locked during access.

1	The number of links to this file
root	The file owner
dev	The group
10876	The size of the file in bytes for the special files, the device driver and number of units
May 16 9:42	The date and time when the file was last changed (modification time)
part2	The filename

1.5.9. BASIC COMMANDS FOR FILE MANUPULATION

ls		list the contents of a directory
	- a	list hidden files
	- l	describes the properties of each file
	- s	give size of each file
file		determine the file type
cat		concatenate and display the file
	- n	precede each line output with its line number
more		browse through a text file
tail		deliver the last part of a file
cp		copy a file
	-i	prompt for confirmation
ln		creating filename aliases (link)
mv		move or rename a file
	-i	prompt for confirmation
rm		remove a file
	-i	prompt for confirmation

1.5.10. BASIC COMMANDS FOR CHANGING FILE ACCESS PERMISSIONS

touch update the access and modification times of a file
chgrp change group, file ownership
chmod change mode, file access permissions (absolute and symbolic mode)

u	user's permissions
g	group's permissions
o	others' permissions
a	all permissions (user, group, and other)

+	Add permissions.
-	Take away permissions.
=	Assign permissions absolutely.

Examples:

```
chmod ugo+r     *
```

Changes user, group, and other to read of all files.

```
chmod go-rwx *
```

Nobody except for the owner can access all files.

Chown - change owner of a file
Chgrp - change the group ownership of a file

1.5.11. BASIC COMMANDS FOR DIRECTORY HANDLING

Protections:	r	Read	Example:	`ls directory_name`
	w	Write	Example:	`rm file`
	x	Execute	Example:	`cd directory_name`

NOTE

Be aware with x and w permision file can be deleted, without file access permissions.

cd	change working directory .. go to the parent directory
cp -r	copy files and subdirectories of a directory (source) to a second directory (destination), along with its files and subdirectory
mkdir	creates a directory -p allows missing parent directories to be created as needed
rmdir	remove a directory (only an empty directory)
rm -r	recursively delete the contents of a directory, its subdirectories, and the directory itself
mv	rename the source directory to the destination directory
pwd	display the pathname of the current working directory

1.6. VI EDITOR

starting vi: `vi filename`

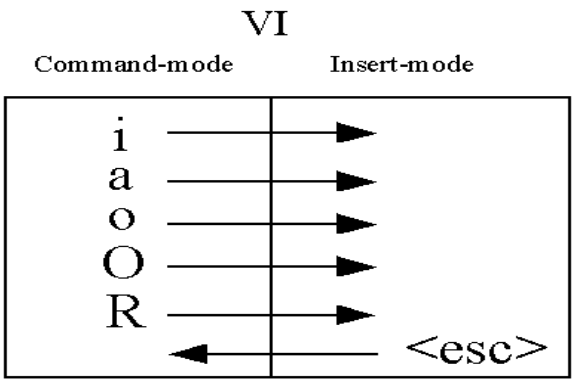
The VI editor has two modes:

The command mode
This is the initial mode upon entering "vi". In this mode the user has capabilities like moving the cursor, save file, and quit file.

stopping vi:	:w	save file
	:w!	overwrite existing file
	:q	quit without saving
	:wq!	save and quit
	:q!	quit without save
moving vi:	h	cursor left
	j	cursor down
	k	cursor up
	l	cursor right
	space	cursor right
	w	word forwards
	b	word backwards
	+	jumps to begin next line
	return	jumps to begin next line
	-	jumps to begin previous line
	^	jumps to begin current line
	$	jumps to end current line
	*n*G	Goto line *n*
	←	scroll cursor left
	↑	scroll cursor up
	→	scroll cursor right
	↓	scroll cursor down

The insert mode
The user must be in this mode to be able to write.

To enter the insert mode:



i	Insert text before cursor
a	Append text after cursor
o	Open a line below cursor
O	Open a line above cursor
R	Replace/type-over a line
<esc>	Back to command mode

To re-enter the command mode simply press the <ESC> key. The screen will blink or beep, when already in this mode.

1.6.1. AN OVERVIEW OF VI EDITOR COMMANDS

/exp	Go forward to exp	hjkl	Unit cursor motion	a	Append after cursor
?exp	Go backward to exp				
		G	Goto last file line	A	Append at line end
n	Repeat previous search	3G	Goto line 3		
N	Reverse previous search	0	Goto line start	c	Change character
		$	Goto line end		
:w	Write buffer to disk				
:w newfile	Write buffer to newfile	H	Goto screen top	cw	Change a word
:w >> file	Append buffer to file	M	Goto screen middle		
:w ! file	Write absolutely	L	Goto screen bottom	c3w	Change 3 words
:q	Quit the editor	w	Go forward 1 word	C	Change line
:q!	Quit, discarding buffer	3w	Go forward 3 words		
:wq	Write buffer and quit	b	Go back 1 word	i	Insert before cursor
:x	Write if needed and quit	3b	Go back 3 words		
:f	Edit & file status			5i	Insert text 5 times
		n	Goto next occurrence		
:r	Read file into buffer	N	Goto previous occurrence	I	Insert at beginning of line
:r file	Read named file in				
:e	(Re)edit file	x	Delete 1 character	o	Open a line below cursor
:e!	(Re)edit file,discard buffer	dw	Delete 1 word		
:e file	Edit named file	dd	Delete 1 line	O	Open a line above cursor
		D	Delete to end of line		
:3,8d	Delete lines 3-8	d0	Delete to begin of line	r	Replace 1 character
:4,9m 12	Move lines 4-9 to 12				
:2,5t 13	Copy lines 2-5 to 13	dG	Delete to end of file	R	Replace/type-over a line
5,9w file	Write lines 5-9 to file	4dd	Delete 4 lines		
				s	Substitute a character
:s/old/new/	Current line, substitute first "old" with "new"	u	Undo last change		
		.	Do last change again	7s	Substitute 7 characters
:s/old/new/g	Current line, substitute each "old" with "new"	Y	Yank a copy of a line	S	Substitute a line
:3,9s/old/new/	Lines 3-9 substitute first "old" with "new"	5Y	Yank a copy of 5 lines		
		p	Put below cursor		
:%s/old/new/	All lines substitute first "old" with "new"	P	Put above cursor		
		J	Join next line to current		
:%s/old/new/g	All lines substitute each "old" with "new"				